支持动态操作的多副本数据完整性验证方案 *

刘洪宇 [a]，丁奕文 [b]，陈雷霆 [a]

(电子科技大学 a. 计算机科学与工程学院; b. 经济与管理学院，成都 611731)

摘　要：针对云存储环境下外包数据面临的安全隐患，并结合现有云数据完整性验证方案的不足，提出了支持动态操作的多副本数据完整性验证方案。方案考虑了多副本应用场景，并在现有云数据完整性验证方案的基础上以较小的代价实现了文件的多副本验证，并通过引入认证的数据结构—基于等级的 Merkle 哈希树，实现了文件的可验证的动态更新。通过对多副本进行关联，可以实现多个副本的同步更新。安全性分析与实验表明了方案的安全性与有效性，方案实现了数据的安全存储与更新，并有效保证了数据多副本的隐私安全。

关键词：多副本；数据完整性验证；动态操作；云存储

中图分类号：TP391　　doi: 10.3969/j.issn.1001-3695.2018.04.0223

Multiple-replica data integrity checking protocol with efficient dynamic update in cloud storage

Liu Hongyu[a], Ding Yiwen[b], Chen Leiting[a]

(*a. School of Computer Science & Engineering, b. School of Economics & Management, University of Electronic Science & Technology of China, Chengdu* 611731, *China*)

Abstract: Considering the security risk of data stored in cloud and the deficiency of existing data integrity checking schemes, this paper proposed a multiple-replica remote data integrity checking protocol with efficient data dynamic update in cloud storage. The scheme was suitable for multiple replica scenario, and achieved multiple-replica provable data possession at a small cost on the basis of existing cloud data integrity checking schemes. By introducing authenticated data structure called rank-based MHT, the protocol supported full data dynamic operations. Through replica correlation, multiple replicas could be updated synchronously. Security analysis and experiment results show the security and effectiveness of the proposed protocol, and can guarantee the privacy of multiple replicas at the same time.

Key words: multiple-replica; data integrity checking; dynamic update; cloud storage

0　引言

云计算作为新的计算模式，使得用户可以按需获取计算资源和存储资源。通过虚拟化技术，用户可以便捷地访问由分布的计算节点组成的共享资源池，节省了硬件设备的购置开销与数据的维护开销。云存储作为云计算最基本、应用最广泛的服务，成为研究人员关注的焦点。数据使用权与管理权相分离所带来的数据安全隐患成为云计算普及的主要障碍之一。高度集中的存储资源使得云服务器面临着严重的安全挑战，外部入侵者的攻击或云服务提供商内部人员的误操作等都使得云中数据的完整性不能得到有效的保证。

数据完整性验证机制使用户可以对存储在云中的数据进行验证并及时发现云数据的损坏情况。目前，根据是否能恢复原始数据，云数据完整性验证协议可以分为数据持有性证明（PDP）协议和可恢复性证明（PoR）协议。前者可检查云数据的完整性而不用下载整个文件，后者在数据完整性验证的基础上允许数据的恢复。这两种协议的主要不同之处是 PoR 机制使用纠错码来进行原始数据的恢复。

Ateniese 等人[1]在 2007 年率先提出数据持有性证明的模型，它利用抽样技术来实现数据的完整性验证，有效减少了用户的计算开销和通信开销。他们还提出了两个基于 RSA 签名的高效且可证明安全的 PDP 方案。然后，出现了基于短签名技术的 PDP 协议[2]，在相同的安全条件下，它拥有更短的签名。此外，这些签名具有同态性质，因此可以将多个签名聚合到一个签名中，短签名 PDP 机制拥有公开验证的特性，用户可以将审计任务交给第三方审计者（TPA），因此，它有更小的存储开销和更低的通信代价。

考虑到用户可能会对存储在云端的数据进行修改，因此，

收稿日期：2018-04-01；修回日期：2018-05-11　　基金项目：新型数据保护密码算法研究（2017YFB0802000）

作者简介：刘洪宇（1976-），男，四川三台人，工程师，博士研究生，主要研究方向为云安全（lhy@uestc.edu.cn）；丁奕文（1994-），女，山东烟台人，硕士，主要研究方向为数据管理；陈雷霆（1966-），男，博士研究生，教授，博导，主要研究方向为云数据与大数据处理.

PDP 方案必须满足动态性。为了解决这一问题，Wang 等人和 Erway 等人[3,4]考虑利用动态的数据结构确保数据块的位置。Wang 等人通过利用经典的 Merkle 哈希树（MHT）改进了可恢复性证明模型。Erway 等人使用了修改的认证跳表（authenticated skip list）数据结构以允许数据动态更新，但它的认证过程较长，需要大量的计算和通信开销。

为了提高数据的可靠性与可用性，在云中存储多个数据副本是被广泛采用的方法。然而云服务提供商可能会为了节省存储空间，获得更多经济利益而不保存数据的副本，使得用户数据的可用性大大受损。Curtmola 等人[5]考虑了存储多个副本时用户数据的完整性验证，它可以以较小的计算开销实现多个副本的验证，但不能支持数据的动态操作。Liu 等人[6]提出了一个基于多副本 Merkle 哈希树的支持动态操作的云数据完整性验证协议，其从上至下产生 Merkle 哈希树的层级，每个数据块的副本块被构建在一个相同的副本子树中。然而它不是针对于分布式存储环境，为了数据的高可用性，副本应被存储于多个地理位置不同的服务器中，否则磁盘的损坏会带来所有副本的破坏；另外，它对每一个副本单独处理，每个副本块有自己的认证标签，会增加用户的计算开销。

近期，随着对云数据完整性协议的广泛研究，涌现了多个 PDP 协议的变体[7~13]，它们实现了比原始 PDP 协议[14~16]更多的特性。2015 年，He 等人[17]考虑了无线体域网环境下的数据完整性验证，并提出了一个无证书的公共审计方案。2016 年，Zhang 等人[18]提出了一个云存储环境下高效的基于身份的公开审计方案。Yu 等人[19]提出了支持完美隐私保护的基于身份的数据完整性验证方案。2017 年，Yan 等人[20]提出了基于同态哈希函数的支持动态操作的远程数据审计协议,但方案并不支持数据的多副本存储。李萌庭等人[21]提出了基于相对索引散列树的数据完整性验证方法。

本文针对多副本场景下的云存储，提出了支持动态操作的多副本数据完整性验证方案。通过对原始文件进行对称加密，生成数据密文，并根据此密文安全生成文件的多个副本，并且以较低的计算开销实现了文件的多副本存储与验证。通过引入认证的数据结构——基于等级的 MHT，实现了文件的可验证的动态更新，包括数据块的修改、增加与删除。对存储在不同云服务器中的副本进行关联，实现了多个副本的同步更新。最后，对方案进行了安全性分析与性能测试，实验结果表明了方案的有效性与实用性。

1 相关知识

1.1 双线性对

G,G_T 是两个阶为素数 p 的乘法循环群，G 为椭圆曲线上的离散点构成的循环群。双线性对 $e:G\times G\to G_T$ 是具有如下性质的映射：

a) 有效可计算性。对所有的 $h_1,h_2\in G$，存在一个有效的计算算法来计算 e。

b) 双线性。对所有的 $h_1,h_2\in G$，$a,b\in Z_p$，有 $e(h_1^a,h_2^b)=e(h_1,h_2)^{ab}$。

c)非退化性。$e(g,g)\neq 1$，其中，g 是 G 的生成元。

1.2 Merkle 哈希树

Merkle 哈希树（MHT）是一个被广泛使用的认证结构，可以被用来有效地检查是否一系列元素被正确存储，因此可用于检查数据的完整性。一个 MHT 是一个二进制树，其中叶子节点包含了所存数据的哈希值，每一个内部节点存储了它的两个孩子的哈希值。每一个内部节点以及根节点由它的两个孩子节点所产生。对每个元素 a_i 而言，其辅助认证信息（AAI）为从 a_i 到达根节点的路径节点的兄弟节点的集合，所以根据元素 a_i 以及其辅助认证信息，可以求出根节点。假定验证者知道 MHT 的根节点，当收到第 i 个数据块 a_i 时，为了检查 a_i 的完整性，除了数据本身，验证者也需得到其辅助认证信息。构建 MHT 使用的单向哈希函数可以保证认证的正确性与可靠性，所以 MHT 可以被用来认证数据块的值，但它不能提供位置的检查。

1.3 基于等级的 Merkle 哈希树

为了使 MHT 可以适用于支持动态操作的数据认证，一种新型的数据结构—基于等级的 MHT（rMHT）被开发出来。在 rMHT 中，除了哈希值 h_v，每一个节点 v 还存储了它能到达的叶子节点的个数。这个值被称作节点 v 的 rank 值，即等级值。可以看出，非叶子节点的等级值等于它的孩子们的叶子节点的等级值的和。不同于 MHT，在 rMHT 中，h_v 的值是将它两个孩子节点的值与它的等级值级联后，再进行哈希。另一个存储在节点中的值是它的高度，用 l_v 表示，最后一个值称为边信息，即是其父亲节点的左孩子还是右孩子，分别用 0 和 1 表示。边信息用 s_v 代表。因此，在 rMHT 中，节点 v 被表示为

$$v=(h_v,r_v,l_v,s_v) \tag{1}$$

更精确地说，假定 v 的孩子节点为 v_l（左孩子节点）和 v_r（右孩子节点），那么节点 v 的值可以被如下计算：

$$l_v=l_{v_l}+1 \tag{2}$$

$$r_v=r_{v_l}+r_{v_r} \tag{3}$$

$$h_v=h(h_{v_l}\,\|\,h_{v_r}\,\|\,r_v) \tag{4}$$

根节点 R 不包含边信息，因此 S_R 为空。在 rMHT 中，第 i 个认证的数据块 a_i 的辅助认证信息为

$$\Omega_{a_i}=\{(v_1,\ldots v_k)\mid v_j=(h_{v_j},r_{v_j},l_{v_j},s_{v_j}),1\le j\le k\} \tag{5}$$

因为辅助认证信息中各节点处在不同的高度，所以要求对于任何 $0\le j<t\le k$，有 $l_{v_j}<l_{v_t}$，意味着辅助认证信息中的的节点按高度从下往上排列，这有利于从底向上有效地计算根节点。图 1 给出了一个构建 rMHT 的例子。对于 $i=\{1,2,\ldots,8\}$，$h_i=H(m_i\|1)$，其中 1 是叶子节点的等级值。$h_c=H(h_1\|h_2\|2)$，$h_d=H(h_3\|h_4\|2)$，$h_e=H(h_5\|h_6\|2)$，$h_f=H(h_7\|h_8\|2)$，其中 2 是 h_c,h_d,h_e,h_f 节点的等级值，因为这些节点都有两个叶子节点。$h_a=H(h_c\|h_d\|4)$，$h_b=H(h_e\|h_f\|4)$，h_a 和 h_b 有 4 个

叶子节点。根节点 h_r 有 8 个叶子节点，所以 $h_r = H(h_a \| h_b \| 8)$ 。

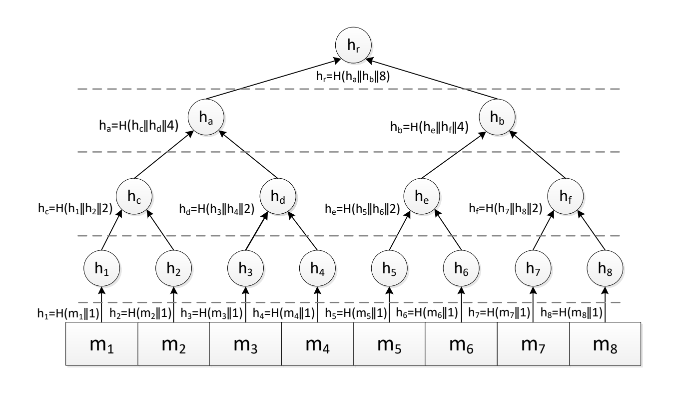


图 1　rMHT 的构建

2　系统模型与设计目标

2.1　系统模型

数据完整性验证的系统模型包含云服务器和数据拥有者两个实体。云服务器有大量的存储资源和强大的计算能力，为数据拥有者提供长期的可扩展的数据存储服务。这里，云服务器包含位于不同地理位置的多个云服务器，数据拥有者可以将文件的多个副本保存在不同的云服务器中。数据拥有者作为资源受限的实体，更倾向于将本地文件存储到云服务器中以节省存储开销，并减轻数据维护与管理负担。为了保证数据的高可用性，数据拥有者选择存储数据的多个副本到云服务器中。

2.2　安全威胁

当数据拥有者将数据存储到云服务器中并删除本地文件后，数据的安全性完全依赖于云服务器。然而存储在云中的数据可能会面临许多不安全因素：首先，数据可能因为云服务提供商内部人员的不当操作或外部攻击者的攻击而遭到丢失或损坏，并且为了声誉，云服务提供商会隐瞒数据丢失事件；其次，云服务提供商可能删除不经常被访问的数据以节省存储空间，并且当用户存储多个文件副本时，云服务器可能会仅存储文件的一个副本，并通过勾结使云用户相信它存储了数据的多个副本。

2.3　设计目标

为了保证数据的安全存储与动态更新，方案应达到以下两个设计目标：

a)文件完整性验证。数据拥有者可以以较小的开销对文件的完整性进行验证。数据拥有者可以进行单副本的验证也可以进行多副本的验证，以确保多个副本的完整性及可用性。

b)可验证的多副本更新。当用户对存储在云中的数据进行修改、增加或删除时，用户可以验证云服务器确实进行了相应的更新。

3　方案构造

3.1　方案概述

一个支持动态操作的多副本 PDP 方案包含 6 种算法，分别是密钥产生、存储、挑战、证据生成、证据验证、可验证的动态更新算法。

密钥产生：数据拥有者运行的概率算法，产生数据拥有者的公钥和私钥。

存储：数据拥有者运行的概率算法，产生文件 F 的 t 个副本。数据拥有者首先对原始文件 F 进行加密产生 F ，对 F 用随机数进行掩蔽以产生多个副本。副本 F_u 将被存储在服务器 S_u 中。然后，数据拥有者产生文件对应的认证标签。数据拥有者首先将文件分成一些块，每个块再划分为不同的扇区，为每个块生成认证标签，之后，数据拥有者为文件构建 rMHT，并对根进行签名。对于文件的所有副本，数据拥有者仅产生一份认证标签。最后，数据拥有者将副本及其认证标签上传到云服务器上，并删除本地副本及标签。云服务器为用户上传的数据建立副本目录[22]，副本目录中包含所有副本的逻辑文件名 LFN 和物理文件名 PFN，LFN 为文件名称 *fname*，各副本拥有相同的 LFN，PFN 由文件存储的物理路径和其云服务器编号构成。

挑战：为了检查云服务器中数据的完整性，数据拥有者向云发起挑战，它可以发起单副本挑战或多副本挑战。对于单副本挑战，数据拥有者与一个特定的云服务器 S_u 进行交互，以确定 S_u 是否完整存储了副本 F_u 。多副本挑战，它可以对存储副本的多个服务器发起挑战，以验证所有副本的可用性。数据拥有者只需要对副本数据块的随机子集进行挑战，就可以以极大概率发现是否存在数据丢失或损坏。

证据生成：由数据拥有者和服务器运行的确定性算法，当数据拥有者对存储在云服务器中的数据进行挑战时，云服务器根据挑战请求生成相应的证据，并将证据发送给数据拥有者。

证据验证：数据拥有者运行的确定性算法，当收到云服务器发送的证据时，数据拥有者对其进行验证以确定存储在云端的数据的完整性。

可验证的动态更新：数据拥有者和云服务器运行的确定性算法，当数据拥有者对云中的数据进行修改、增加或删除时，云服务器根据更新请求对数据的所有副本进行操作，并返回更新的证据，使数据拥有者可以对更新进行验证。

3.2　具体构造

$e: G \times G \rightarrow G_T$ 是一个双线性映射， G ， G_T 是两个乘法循环群， $ord(G) = ord(G_T) = p$ ， p 是一个大素数， g 是 G 的生成元。 $H: \{0,1\}^* \rightarrow G$ 是一个抗碰撞的哈希函数，用于构建 rMHT。 φ 是一个安全的伪随机函数。

密钥产生：数据拥有者首先在 Z_p 中随机选择 α ，并计算 $v \leftarrow g^{\alpha}$ ，然后产生一个随机的签名密钥对 $(spk, ssk) \xleftarrow{R} SKg$ ，并产生 AES 加密密钥 K 和伪随机函数密钥 z。数据拥有者的私钥为 $sk = (\alpha, ssk, K, z)$ ，公钥为 $pk = (v, spk)$ 。

存储：数据拥有者首先将原始文件 F 分成 n 个块，即 $F=\{f_1,f_2,\dots f_n\}$，用密钥 K 对原始文件 F 进行加密获得 $F=\{b_1,b_2,\dots b_n\}$，其中 $b_i=E_K(f_i)$，$1\le i\le n$。然后，数据拥有者如下产生 t 个不同的文件副本：对于副本 F_u，$1\le u\le t$，调用伪随机函数 φ，生成随机的 r_u，$r_u=\varphi_z(u)$。F_u 中的数据块由 F 的每个数据块与随机值 r_u 运算产生，即 $F_u=(m_{u,1},\dots,m_{u,n})$，其中，$m_{u,i}$=$b_i+r_u$，$1\le i\le n$。数据拥有者将文件 F 的每个数据块分为 s 个扇区，每个扇区表示为 $\{b_{i,j}\}$，$1\le i\le n$，$1\le j\le s$。数据拥有者从 Z_p 中为文件随机选取一个文件名 $fname$，并从 G 中选取 s 个随机元素 $u_1,\dots u_s\leftarrow G$，然后为文件 F 的每个块产生认证标签 T_i，并生成标签集合 $T=\{T_1,T_2,\dots T_n\}$，其中：

$$T_i=(H(fname)\cdot\prod_{j=1}^{s}u_j^{\,b_{i,j}})^{\alpha} \tag{6}$$

对于文件 F，t_0 表示 $fname\|n\|u_1\|\cdots\|u_s$，文件标签 t 为 t_0 和它的签名的级联，签名私钥为 ssk，即 $t\leftarrow t_0\|SSig_{ssk}(t_0)$。数据拥有者将副本 F_u，标签集合 T 以及文件标签 t 存储在服务器 S_u 上，然后，数据拥有者删除原始文件及其各副本以及标签集合、文件标签。云服务器为用户上传的数据建立副本目录，它包含所有副本的逻辑文件名 LFN 和物理文件名 PFN。LFN 为文件名称 $fname$，各副本拥有相同的 LFN，PFN 由文件存储的物理路径和其云服务器编号构成，并将副本目录发送给数据拥有者。

挑战：数据拥有者随机选择 l 个 $\{(i,v_i)\}$ 组成挑战集合 Q，其中，$i\in[1,n]$，$v_i\in Z_p$，数据拥有者将针对 F_u 的挑战 $\{Q,fname\}$ 发送给相应的云服务器。

证据生成：收到挑战后，云服务器根据 Q 中的 $\{(i,v_i)\}$ 计算

$$\mu_j\leftarrow\sum_{(i,v_i)\in Q}v_i m_{i,j}\,,\quad 1\le j\le s \tag{7}$$

$$\sigma\leftarrow\prod_{(i,v_i)\in Q}T_i^{v_i} \tag{8}$$

然后，云服务器将 $\mu_1,\dots\mu_s$，σ，t 发送给数据拥有者。

证据验证：数据拥有者首先验证文件标签 t 中的签名，若签名无效，则终止；否则，从 t 中得到 $u_1,\dots,u_s$，并验证：

$$e(\sigma,g)=e(H(fname)^{\sum_{(i,v_i)\in Q}v_i}\cdot\prod_{j=1}^{s}u_j^{\delta_j},v) \tag{9}$$

$$\delta_j=\mu_j-(\sum_{(i,v_i)\in Q}v_i)r_u \tag{10}$$

是否成立。

同时，数据拥有者通过 rMHT 检验相应挑战块的位置。通过验证一个节点的辅助认证信息中边信息为左的节点的等级值的和可以判断节点的位置是否正确。将 CoR 记为待验证节点的左侧的叶子节点（包含它自己）的个数。例如，在图 1 中，如果数据拥有者想验证数据块 m_4 的值和位置，它从云服务器端收到辅助认证信息 $\Omega_{a_4}=\{v_1,v_2,v_3\}=\{(h_3,1,0,0),\ a_4$。然后，它将 CoR 设为 1，并进行如下验证过程：

a) 计算节点 d 的等级值 r_d =1+1=2，并计算 d 的值 $h_d=h(h_3\|h(a_4\|1)\|2)$，CoR 的值为 CoR=1+1=2；

b) 计算点 a 的等级值 $r_a=2+2=4$，a 的值为 $h_a=h(h_c\|h_d\|4)$，此时 CoR 的值为 CoR=2+2=4；

c) 计算根节点 $h_R^{'}$ 的等级值为 4+4=8，$h_R^{'}$ 的值为 $h(h_a\|h_b\|8)$；

d) 检查是否 $h_R^{'}=h_R$ 并验证是否 CoR=4.

当验证等式(9)成立且挑战节点的位置均正确时，说明保存在云服务器中的数据是完整的。

可验证的动态更新：这里考虑常见的三种动态操作，即修改、增加和删除数据块。

修改操作指将某一特定数据块替换为新的数据块。对于修改操作，云服务器将数据块更新后，还需对 rMHT 进行更新。假设数据拥有者欲将第 i 个数据块更新为 a_i^*，它将 (Mod,i,a_i^*,T_i^*) 发送给云服务器，云服务器首先更新第 i 个节点的哈希值，然后更新从改叶子节点到根节点的路径上各节点的值，向数据拥有者返回新的根值 h_R^* 和 a_i^* 的新的路径节点 $P_{a_i^*}^*$。

增加操作指将一个新数据块插入到原来的数据块集合中。数据拥有者向云服务器发送 (Ins,i,a_i^*,T_i^*)，表明将数据块 a_i^* 插入到第 i 个位置上，即 a_i 之前。对于增加操作，云服务器对 rMHT 的更新过程如下：计算新节点的哈希值，并设置它的等级值；计算 a_i^* 和 a_i 的父节点，并且对于 a_i 的辅助认证信息 AAI 中节点的兄弟节点，重新计算它们节点的值；最后，输出新的根节点的值 h_R^* 和 a_i^* 的路径节点 $P_{a_i^*}^*$。

删除操作指将一个特定数据块从原先数据集中移除。数据拥有者向云服务器发送 (Del,i)，表明删除第 i 个数据块。对于删除操作，云服务器更新 rMHT 的过程如下：将 a_i 的所有路径节点更新，将待删节点的兄弟节点的层值加 1，并更新其边信息；更新 a_i 的辅助认证信息 AAI 中节点的兄弟节点；最后，输出新的根节点的值 h_R^* 和 a_{i-1} 新的路径节点 $P_{a_{i-1}}^*$。

为了验证服务器已经按照要求进行了更新操作，数据拥有者向云服务器取回更新算法必要的输入。例如节点的辅助认证信息 AAI，验证其正确性后，运行更新算法，得到新的根值 h_R，与从服务器端发来的更新后的根值 h_R^* 进行比较。若相同，则说明云服务器已按要求对数据进行了更新。

当数据拥有者对文件的所有副本进行动态更新时，向其中一个存储副本的服务器发送针对所有副本的更新命令。例如，对于修改操作，数据拥有者向云服务器发送 (all,Mod,i,a_i^*,T_i^*)，此云服务器将会向副本目录中存储此文件的所有物理地址发送数据修改命令，各服务器将根据命令进行修改。数据拥有者可随机与某一云服务器进行交互，对其更新操作进行验证。

4　方案分析

4.1　功能性分析

本文所提方案适用于云用户将文件的多个副本存储在云服务器上，云用户只用为文件生成一份标签，便可以通过此标签对存储在各个服务器的不同的文件副本进行完整性验证，而不用为每个文件单独生成标签，同时不用担心云服务器只存储了一份文件副本以欺骗用户。云用户可以通过与多个服务器进行交互以验证各个副本的完整性。

当用户对文件进行动态更新时，向存储副本的云服务器发送动态操作请求，云服务器对文件进行更新后，将生成的证据发送给云用户，云用户可以对云服务器的动态操作进行验证，实现了可验证的动态更新。同时，由于云服务器上存储了副本目录，它向其他存储文件副本的多个云服务器发送数据更新请求，其他云服务器将依照请求对文件进行更新，云用户可以通过与存储副本的云服务器进行交互以验证数据的动态更新。

4.2　安全性分析

在本文的安全假设中，数据拥有者不会主动泄露密钥信息，并且默认产生副本时采用的是安全的伪随机函数。数据拥有者对原始文件采用的对称加密算法为 AES 加密算法，其算法的攻击复杂度依赖于分组长度和密钥长度，它同时兼具了高效率与高安全性。文件的副本的安全性基于 AES 加密算法以及伪随机函数 φ 。

对于文件的完整性验证过程，文献[1]证明了如果生成文件标签所采用的的数字签名方案是存在性不可伪造的，双线性群中的计算 Diffie-Hellman 问题是困难的，则在随机预言机模型中，没有多项式时间敌手能够以不可忽略的概率攻破方案的可靠性，即敌手能伪造出证据，它能够通过数据拥有者的验证。具体证明这里不再赘述。通过使用 $\{H(m_i\|1),\Omega_i\}_{i\in Q}$ 和 $sig_{ssk}(h_R)$，数据拥有者可以通过验证 h_R 的有效性以及辅助认证信息 Ω 的正确性。通过 Ω 以及挑战的数据块，数据拥有者可以计算出数据块的位置信息 CoR，从而保证数据位置的正确性，因此可以抵抗替换攻击。通过数据块的值与位置的正确性的保证，数据拥有者可以确信保存在云服务器中的数据是完整的，敌手不能在多项式时间内伪造出能通过数据拥有者验证的证据，从而保证了方案的安全性。

5　实现结果与分析

本章将通过实验测试本文提出的方案的性能。对于测试平台，本文采用了 Intel Core(TM) i7-7500 @2.70 GHz 处理器，8 GB 内存，Win10 64 位操作系统。算法由 Visual Studio 2012 编译实现，并调用了 Miracl 库，其被公认为是椭圆曲线密码学的黄金开源标准。在测试中，安全参数被设为 λ=80，满足了安全性要求。下面将从两个方面说明算法的测试结果。

a)本文测试了标签生成（TagGen）的时间开销，模拟了当文件大小从 1 KB 逐渐增加到 10 KB 时，数据拥有者生成标签所需花费的时间。其中，每个扇区大小为 160 比特，每次使文件大小增加 1 KB。根据标签生成算法，当文件大小固定时，每个数据块中包含的扇区数越多，数据块个数越少，因此总的生成标签时间基本不变；而当文件大小逐渐增加时，总的扇区数随之增加，标签生成时间也会逐渐增加。文件大小增大时标签生成时间如图 2 所示。由图 2 可以看出，标签生成的时间消耗随着文件大小呈现线性增长，与数值分析结果一致；当文件大小为 10KB 时，用户生成标签时间大约为 0.95 s。

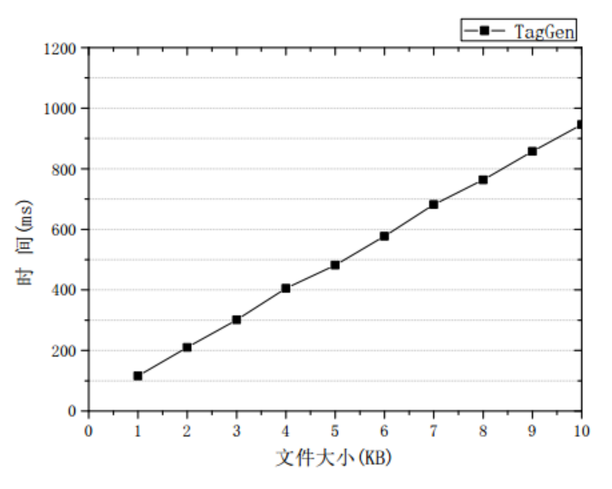


图 2　文件大小增大时标签生成时间

b)本文测试了完整性验证过程中数据拥有者和云服务器的性能开销。将文件大小固定为 1 MB，并逐步增加数据拥有者挑战的块的个数，从 50 递增到 500。挑战的数据块增加时证据生成及验证时间如图 3 所示。从图 3 可以看出，当挑战块数逐渐增加时，云服务器的计算开销逐步增大，从 16 ms 逐渐增加到 165 ms，数据拥有者的验证开销基本保持在 193 ms，这是因为随着挑战块个数的增加，服务器生成证据 $\mu_j, 1\le j\le s$ 和 σ 的时间在逐渐增加，而数据拥有者计算验证等式所需的时间仅随着挑战块个数的递增有微小的增加。Ateniese 等人证明如果云服务器中有 1%的数据损坏，则数据拥有者通过挑战 300 个块可以以 95%的概率检测出数据损坏，挑战 460 个块可以以 99%的概率检测出数据损坏。从图中可以看出，当挑战 300 个块时，云服务器生成证据所需花费时间为 99 ms，挑战 460 个块时，花费时间为 152 ms，数据拥有者验证时间均为 194 ms。

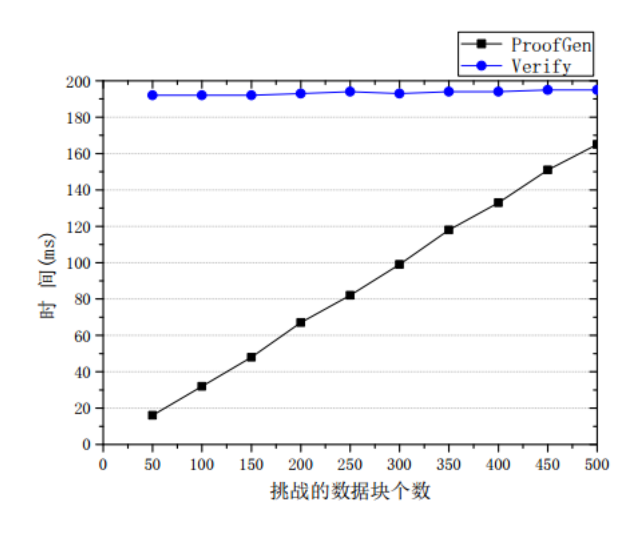


图 3　挑战的数据块增加时证据生成及验证时间

6　结束语

本文设计了一种云存储环境下的支持动态操作的多副本数据完整性验证方案，能够使数据拥有者以较低的计算开销实现多个副本的存储与完整性验证；同时支持数据的动态操作，利用认证的数据结构 rMHT 保证了数据块的位置的正确性，并实现可验证的动态更新。安全性分析表明了方案满足了安全要求，实验结果表明了方案的高效性与实用性。下一步研究工作是设计一个云环境下支持公开验证的多副本数据完整性验证方案，同时支持可验证的动态操作，并尝试进一步提升计算效率，减少通信开销，探索出更高效，更灵活的完整性验证方案。

参考文献：

[1] Ateniese G, Burns R C, Curtmola R, *et al.* Provable data possession at untrusted stores [C]// Proc of ACM CCS'07. 2007: 598-609.

[2] Dan B, Lynn B, Shacham H. Short signatures from the weil pairing [J]. Journal of Cryptology, 2004, 17 (4): 297-319.

[3] Wang Qian, Wang Cong, Ren Kui, *et al.* Enabling public auditability and data dynamics for storage security in cloud computing [C]// Proc of ESORICS'09. 2009: 355-370.

[4] Erway C, Alptekin K, Papamanthouc C. Dynamic provable data possession [C]// Proc of ACM Conference on Computer and Communications Security. 2009: 213-222.

[5] Curtmola R, Khan O, Burns R, *et al.* MR-PDP: multiple-replica provable data possession [C]// Proc of IEEE International Conference on Distributed Computing Systems. 2008: 411-420.

[6] Liu Chang, Ranjan R, Yang Chi, *et al.* MuR-DPA: top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud [J]. IEEE Trans on Computers, 2015, 64 (9): 2609-2622.

[7] Wang Qian, Wang Cong, Ren Kui, *et al.* Enabling public auditability and data dynamics for storage security in cloud computing [J]. IEEE Trans on Parallel and Distributed Systems, 2011, 22 (5): 847-859.

[8] Wang Cong, Ren Kui, Lou Wenjing, *et al.* Toward public auditable secure cloud data storage services [J]. Network IEEE, 2010, 24 (4): 19-24.

[9] Zhu Yan, Hu Hongxin, Ahn G J, *et al.* Efficient audit service outsourcing for data integrity in clouds [J]. Journal of System and Software, 2012, 85 (5): 1083-1095.

[10] Zhu Yan, Hu Hongxin, Ahn G J, *et al.* Cooperative provable data possession for integrity verification in multicloud storage [J]. IEEE Trans on Parallel and Distributed Systems, 2012, 23 (12): 2231-2244.

[11] Yang Kan, Jia Xiaohua. An efficient and secure dynamic auditing protocol for data storage in cloud computing [J]. IEEE Trans on Parallel and Distributed Systems, 2013, 24 (9): 1717-1726.

[12] Zhu Yan, Wang Shangbiao, Hu Hongxin, *et al.* Secure collaborative integrity verification for hybrid cloud environments [J]. International Journal of Cooperative Information Systems, 2012, 21 (3): 165-198

[13] Wang Cong, Chow S S M, Wang Qian, *et al.* Privacy-preserving public auditing for secure cloud storage [J]. IEEE Trans on Computers, 2013, 62 (2): 362-375.

[14] Juels A, Kaliski B S. PORs: proofs of retrievability for large files [C]// Proc of ACM CCS'07. 2007: 584-597.

[15] Shacham H, Waters B. Compact proofs of retrievability [C]// Proc of Asiacrypt'08. 2008: 90-107.

[16] Shacham H, Waters B. Compact proofs of retrievability [J]. Journal of Cryptology, 2013, 26 (3): 442-483.

[17] He Debiao, Zeadally S, Wu Libing. Certificateless public auditing scheme for cloud-assisted wireless body area networks [J]. IEEE Systems Journal, 2015, PP (99): 1-10.

[18] Zhang Jianhong, Dong Qiaocui. Efficient ID-based public auditing for the outsourced data in cloud storage [J]. Journal of Clinical Ultrasound, 2016, 6 (6): 1-14

[19] Yu Yong, Man H A A, Ateniese G, *et al.* Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage [J]. IEEE Trans on Information Forensics & Security, 2017, PP (99): 1-1.

[20] Yan Hao, Li Jiguo, Han Jinguang, *et al.* A novel efficient remote data possession checking protocol in cloud storage [J]. IEEE Transon Information Forensics & Security, 2017, 12 (1): 78-88.

[21] 李萌庭，周安宁. 云环境中基于相对索引散列树的数据审核方法 [J/OL]. 计算机应用研究，2019, 36 (4) . [2018-02-07]. http://www.arocmag. com/article/02-2019-04-039. html.

[22] 田荣阳. 数据网格中的副本定位及选择服务 [D]. 重庆：重庆大学，2006.